The "yuima" package : an R framework for simulation and inference of stochastic differential equations (Frontiers in mathematical science through collaborations with other disciplines)

AUTHOR(S):

Iacus, Stefano Maria

CITATION:

Iacus, Stefano Maria. The "yuima" package : an R framework for simulation and inference of stochastic differential equations (Frontiers in mathematical science through collaborations with other disciplines). 数理解析研究所講究録 2011, 1752: 85-112

ISSUE DATE:

2011-07

URL:

http://hdl.handle.net/2433/171154

数理解析研究所講究録
第1752巻 2011年 85-112

The "**yuima**" package: an R framework for simulation and inference of stochastic differential equations

Stefano Maria Iacus
Department of Economics, Business and Statistics
University of Milan
Via Conservatorio 7, 20124 Milan, Italy

Abstract

The Yuima Project is an open source and collaborative effort of several mathematicians and statisticians aimed at developing the R package named "**yuima**" for simulation and inference of stochastic differential equations.

In the **yuima** package stochastic differential equations can be of very abstract type, e.g. uni or multidimensional, driven by Wiener process of fractional Brownian motion with general Hurst parameter, with or without jumps specified as Lévy noise. Lévy processes can be specified via compound Poisson description, by the specification of the Lévy measure or via increments and stable laws.

The **yuima** package is intended to offer the basic infrastructure on which complex models and inference procedures can be built on. In particular, the basic set of functions includes the following: i) simulation schemes for all types of stochastic differential equations (Wiener, fBm, Lévy); ii) different subsampling schemes including random sampling with user specified random times distribution, space discretization, tick times, etc. iii) automatic asymptotic expansion for the approximation and estimation of functionals of diffusion processes with small noise via Malliavin calculus, useful in option pricing; iv) efficient quasi-likelihood inference for diffusion processes and model selection.

1 Introduction

The YUIMA Project[1] is an open source[2] academic project aimed at developing the R package named "**yuima**" for simulation and inference of stochastic differential equations. The YUIMA Project is mainly developed by mathematicians and statisticians who actively publish in the field of inference and simulation for stochastic differential equations. The YUIMA Project Core Team, currently consists of the following people: A. Brouste, M. Fukasawa, H. Hino, S.M. Iacus, K. Kamatani, H.Masuda, Y. Shimizu, M. Uchida, N. Yoshida.

The **yuima** package provides an object-oriented programming environment for simulation and statistical inference for stochastic processes by R. The **yuima** package adopts the S4 system of classes and methods (Chambers, 1998).

Under this framework, the **yuima** package also supplies various functions to execute simulation and statistical analysis. Both categories of procedures may depend each other. Statistical inference often requires a simulation technique as a subroutine, and a certain simulation method needs to fix a tuning parameter by applying a statistical methodology. It is especially the case of stochastic processes because most of expected values involved do not admit an explicit expression. The **yuima** package facilitates comprehensive, systematic approaches to the solution.

Stochastic differential equations are commonly used to model random evolution along continuous or practically continuous time, such as the random movements of a stock price. Theory of statistical inference for stochastic differential equations already has a fairly long history, more than three decades, but it is still developing quickly new methodologies and expanding the area. The formulas produced by the theory are usually very sophisticated, which makes it difficult for standard users not necessarily familiar with this field to enjoy utilities. For example, the asymptotic expansion method for computing option prices (i.e., expectation of an irregular functional of a stochastic process) provides precise approximation values instantaneously, taking advantage of the analytic approach, but the formula has a long expression like more than one page!

The **yuima** package delivers up-to-date methods as a package onto the desk of the user working with simulation and/or statistics for stochastic differential equations.

Sampled data from a continuous-time process features the time stamps as well as the positions of the object. It is requiring a new theory of estimation.

[1]The Project has been funded up to 2010 by the Japan Science Technology (JST) Basic Research Programs PRESTO, Grants-in-Aid for Scientific Research No. 19340021.

[2]All code in the **yuima** package is subject to the GNU General Public License, Version 2, see `http://www.gnu.org/licenses/gpl-2.0.html`.

The **yuima** framework can apply multi-dimensional time stamps of tick data and provides diverse functions handling such kind data to support statistical analysis.

Although we assume that the reader of this paper has a basic knowledge of the R language, most of the examples are easy to be understood by anyone.

2 The yuima package

The package **yuima** depends on some other packages, like **zoo**, which can be installed separately. The package **zoo** is used internally to store time series data. This dependence may change in the future adopting a more flexible class for internal storage of time series.

2.1 How to obtain the package

The **yuima** package is hosted on R-Forge and the web page of the Project is `http://r-forge.r-project.org/projects/yuima`. The R-Forge page contains the latest development version, and stable version of the package as also available through CRAN. Development versions of the package are not supposed to be stable or functional, thus the occasional user should consider to install the stable version first. The package can be installed from R-Forge using `install.packages("yuima", repos = "http://R-Forge.R-project.org")` and for the CRAN version, via `install.packages("yuima")`.

2.2 The main object and classes

Before discussing the methods for simulation and inference for stochastic processes solutions to stochastic differential equations, here we discuss the main classes in the package. As mentioned there are different classes of object defined in the **yuima** package and the main class is called the `yuima-class` and it is composed of several slots. Figure 1 represents the different classes and their slots. The different slots do not need to be all present at the same time. For example, in case one wants to simulate a stochastic process, only the slots `model` and `sampling` should be present, while the slot `data` will be filled by the simulator. We now discuss in details the different object separately.

2.3 The `yuima.model` class

In **yuima** three main classes of stochastic differential equations can be easily specified. All multidimensional and eventually as parametric models.

- diffusions $\mathrm{d}X_t = a(t, X_t)dt + b(t, X_t)\mathrm{d}W_t$, where W_t is a standard Brownian motion;

- fractional Gaussian noise, with H the Hurst parameter

$$\mathrm{d}X_t = a(t, X_t)dt + b(t, X_t)\mathrm{d}W_t^H;$$

- diffusions with jumps and Lévy processes solution to

$$\begin{aligned}\mathrm{d}X_t = \; & a(X_t)dt + b(X_t)\mathrm{d}W_t + \int_{|z|>1} c(X_{t-}, z)\mu(\mathrm{d}t, \mathrm{d}z) \\ & + \int_{0<|z|\leq 1} c(X_{t-}, z)\{\mu(\mathrm{d}t, \mathrm{d}z) - \nu(\mathrm{d}z)\mathrm{d}t\}.\end{aligned}$$

The `yuima.model` class contains informations about the stochastic differential equation of interest. The constructor `setModel` is used to give a mathematical description of the stochastic differential equation. All functions in the package are assumed to get as much information as possible from the model instead of replicating the same code everywhere. If there are missing pieces of information, we may change or extend the description of the model.

An object of `yuima.model` contains several slots listed below. To see inside its structure, we use the R command `str`.

- `drift` is an R expression which contains the drift specification.
- `diffusion` is itself a list of 1 slot which describes the diffusion coefficient relative to first noise.
- `parameter` which is a short name for "parameters" which is a list of objects.
- `all` contains the names of all the parameters found in the diffusion and drift coefficient.
- `common` contains the names of the parameters in common between the drift and diffusion coefficients.
- `diffusion` contains the parameters belonging to the diffusion coefficient.
- `drift` contains the parameters belonging to the drift coefficient.
- `solve.variable` contains a vector of variable names, each element corresponds to the name of the solution variable (left-hand-side) of each equation in the model, in the corresponding order.

- `state.variable` and `time.variable`, by default, are assumed to be x and t but the user can freely choose them. The `yuima.model` function assumes that the user either use default names for `state.variable` and `time.variable` variables or specify his own names. All the rest of the symbols are considered parameters and distributed accordingly in the `parameter` slot.

- `noise.number` indicates the number of sources of noise.

- `equation.number` represents the number of equations, i.e. the number of one dimensional stochastic differential equations.

- `dimension` reports the dimensions of the parameter space. It is a list of the same length of `parameter` with the same names.

In order to show how general is the approach in the **yuima** package we present some examples.

2.3.1 Diffusion processes

Assume that we want to describe the following stochastic differential equation

$$\mathrm{d}X_t = -3X_t\mathrm{d}t + \frac{1}{1+X_t^2}\mathrm{d}W_t$$

This is done in **yuima** specifying the drift and diffusion coefficients as plain mathematical expressions

```
R> mod1 <- setModel(drift = "-3*x",
+     diffusion = "1/(1+x^2)")
```

At this point, the package fills the proper slots of the yuima object

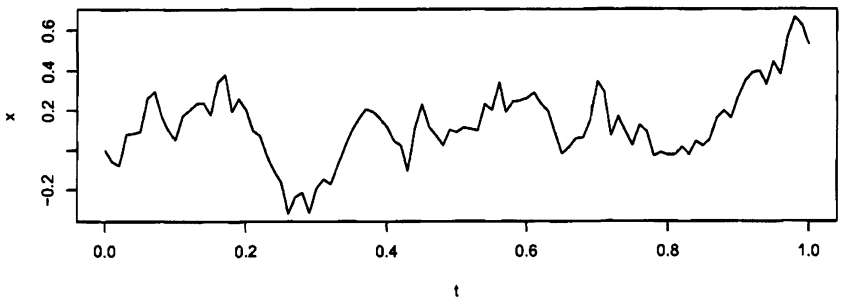
```
R> str(mod1)

Formal class 'yuima.model' [package "yuima"] with 16 slots
  ..@ drift          :  expression((-3 * x))
  ..@ diffusion      :List of 1
  .. ..$ :  expression(1/(1 + x^2))
  ..@ hurst          : num 0.5
  ..@ jump.coeff     :  expression()
  ..@ measure        : list()
  ..@ measure.type   : chr(0)
  ..@ parameter      :Formal class 'model.parameter' [package "yuima"] with 6 slots
  .. .. ..@ all      : chr(0)
  .. .. ..@ common   : chr(0)
  .. .. ..@ diffusion: chr(0)
  .. .. ..@ drift    : chr(0)
  .. .. ..@ jump     : chr(0)
  .. .. ..@ measure  : chr(0)
  ..@ state.variable : chr "x"
  ..@ jump.variable  : chr(0)
  ..@ time.variable  : chr "t"
  ..@ noise.number   : num 1
```

```
  ..@ equation.number: int 1
  ..@ dimension      : int [1:6] 0 0 0 0 0 0
  ..@ solve.variable : chr "x"
  ..@ xinit          : num 0
  ..@ J.flag         : logi FALSE
```

And it is possible to see that the jump coefficient is void and the Hurst parameter is set to 0.5, because this corresponds to the standard Brownian motion. Now, with `mod1` in hands, it is very easy to simulate a trajectory of the process as follows

```
R> set.seed(123)
R> X <- simulate(mod1)
R> plot(X)
```



The `simulate` function fills in addition the two slots `data` and `sampling` of the `yuima` object.

```
R> str(X, vec.len = 2)

Formal class 'yuima' [package "yuima"] with 5 slots
  ..@ data          :Formal class 'yuima.data' [package "yuima"] with 2 slots
  .. .. ..@ original.data: ts [1:101, 1] 0 -0.056 ...
  .. .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. .. ..$ : NULL
  .. .. .. .. ..$ : chr "Series 1"
  .. .. .. ..- attr(*, "tsp")= num [1:3] 0 1 100
  .. .. ..@ zoo.data     :List of 1
  .. .. .. ..$ Series 1:âÃŸzooregâÃŽ series from 0 to 1
  Data: num [1:101] 0 -0.056 ...
  Index:  num [1:101] 0 0.01 0.02 0.03 0.04 ...
  Frequency: 100
  ..@ model         :Formal class 'yuima.model' [package "yuima"] with 16 slots
  .. .. ..@ drift          :  expression((-3 * x))
  .. .. ..@ diffusion      :List of 1
  .. .. .. ..$ :  expression(1/(1 + x^2))
  .. .. ..@ hurst          : num 0.5
  .. .. ..@ jump.coeff     :  expression()
  .. .. ..@ measure        : list()
  .. .. ..@ measure.type   : chr(0)
  .. .. ..@ parameter      :Formal class 'model.parameter' [package "yuima"] with 6 slots
  .. .. .. .. ..@ all      : chr(0)
  .. .. .. .. ..@ common   : chr(0)
  .. .. .. .. ..@ diffusion: chr(0)
  .. .. .. .. ..@ drift    : chr(0)
  .. .. .. .. ..@ jump     : chr(0)
```

```
.. .. .. .. ..@ measure  : chr(0)
.. .. ..@ state.variable : chr "x"
.. .. ..@ jump.variable  : chr(0)
.. .. ..@ time.variable  : chr "t"
.. .. ..@ noise.number   : num 1
.. .. ..@ equation.number: int 1
.. .. ..@ dimension      : int [1:6] 0 0 0 0 0 ...
.. .. ..@ solve.variable : chr "x"
.. .. ..@ xinit          : num 0
.. .. ..@ J.flag         : logi FALSE
..@ sampling      :Formal class 'yuima.sampling' [package "yuima"] with 11 slots
.. .. ..@ Initial      : num 0
.. .. ..@ Terminal     : num 1
.. .. ..@ n            : num 100
.. .. ..@ delta        : num 0.01
.. .. ..@ grid         :List of 1
.. .. .. ..$ : num [1:101] 0 0.01 0.02 0.03 0.04 ...
.. .. ..@ random       : logi FALSE
.. .. ..@ regular      : logi TRUE
.. .. ..@ sdelta       : num(0)
.. .. ..@ sgrid        : num(0)
.. .. ..@ oindex       : num(0)
.. .. ..@ interpolation: chr "pt"
..@ characteristic:Formal class 'yuima.characteristic' [package "yuima"] with 2 slots
.. .. ..@ equation.number: int 1
.. .. ..@ time.scale     : num 1
..@ functional    :Formal class 'yuima.functional' [package "yuima"] with 4 slots
.. .. ..@ F    : NULL
.. .. ..@ f    : list()
.. .. ..@ xinit: num(0)
.. .. ..@ e    : num(0)
```

2.3.2 Parametric models

When a parametric model like

$$\mathrm{d}X_t = -\theta X_t \mathrm{d}t + \frac{1}{1+X_t^\gamma}\mathrm{d}W_t$$

is specified, **yuima** attempts to distinguish the parameters' names from the ones of the state and time variables

```
R> mod2 <- setModel(drift = "-theta*x",
+      diffusion = "1/(1+x^gamma)")
```

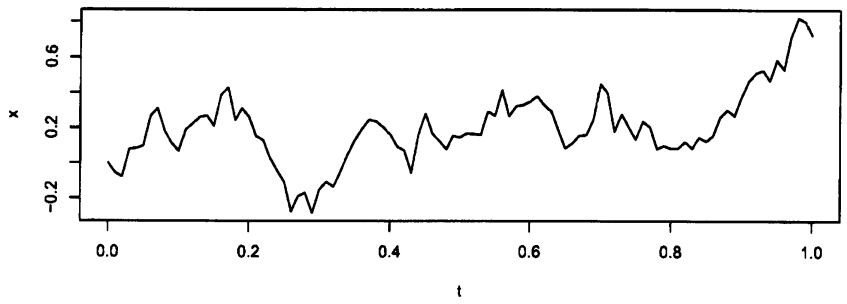
```
R> str(mod2)
```

```
Formal class 'yuima.model' [package "yuima"] with 16 slots
  ..@ drift          :  expression((-theta * x))
  ..@ diffusion      :List of 1
  .. ..$ :  expression(1/(1 + x^gamma))
  ..@ hurst          : num 0.5
  ..@ jump.coeff     :  expression()
  ..@ measure        : list()
  ..@ measure.type   : chr(0)
  ..@ parameter      :Formal class 'model.parameter' [package "yuima"] with 6 slots
  .. .. ..@ all      : chr [1:2] "theta" "gamma"
  .. .. ..@ common   : chr(0)
  .. .. ..@ diffusion: chr "gamma"
  .. .. ..@ drift    : chr "theta"
```

```
.. .. ..@ jump      : chr(0)
.. .. ..@ measure   : chr(0)
..@ state.variable : chr "x"
..@ jump.variable  : chr(0)
..@ time.variable  : chr "t"
..@ noise.number   : num 1
..@ equation.number: int 1
..@ dimension      : int [1:6] 2 0 1 1 0 0
..@ solve.variable : chr "x"
..@ xinit          : num 0
..@ J.flag         : logi FALSE
```

In order to simulate the parametric model it is necessary to specify the values of the parameters as the next code shows

```
R> set.seed(123)
R> X <- simulate(mod2, true.param = list(theta = 1,
+     gamma = 3))
R> plot(X)
```



2.3.3 Multidimensional processes

Next is an example with two stochastic differential equations driven by three independent Brownian motions

$$\begin{aligned} \mathrm{d}X_t^1 &= -3X_t^1\mathrm{d}t + \mathrm{d}W_t^1 + X_t^2\mathrm{d}W_t^3 \\ \mathrm{d}X_t^2 &= -(X_t^1 + 2X_t^2)\mathrm{d}t + X_t^1\mathrm{d}W_t^1 + 3\mathrm{d}W_t^2 \end{aligned}$$

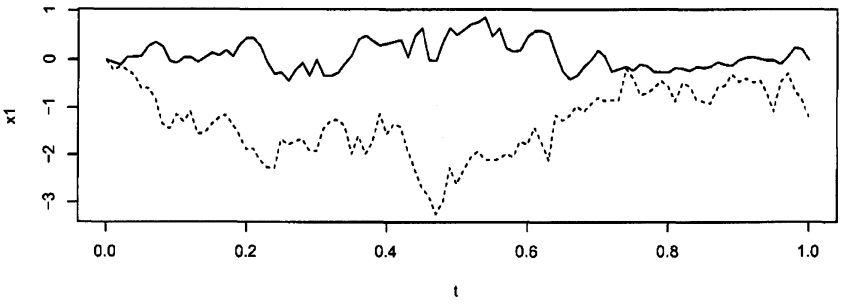
but this has to be organized into matrix form

$$\begin{pmatrix} \mathrm{d}X_t^1 \\ \mathrm{d}X_t^2 \end{pmatrix} = \begin{pmatrix} -3X_t^1 \\ -X_t^1 - 2X_t^2 \end{pmatrix} \mathrm{d}t + \begin{bmatrix} 1 & 0 & X_t^2 \\ X_t^1 & 3 & 0 \end{bmatrix} \begin{pmatrix} \mathrm{d}W_t^1 \\ \mathrm{d}W_t^2 \\ \mathrm{d}W_t^3 \end{pmatrix}$$

```
R> sol <- c("x1", "x2")
R> a <- c("-3*x1", "-x1-2*x2")
R> b <- matrix(c("1", "x1", "0", "3",
+     "x2", "0"), 2, 3)
R> mod3 <- setModel(drift = a, diffusion = b,
+     solve.variable = sol)
```

Again, this model can be easily simulated

```
R> set.seed(123)
R> X <- simulate(mod3)
R> plot(X, plot.type = "single", lty = 1:2)
```



But it is also possible to specify more complex models like the following

$$\begin{cases} \mathrm{d}X_t^1 = X_t^2 \, |X_t^1|^{2/3} \, \mathrm{d}W_t^1, \\ \mathrm{d}X_t^2 = g(t)\mathrm{d}X_t^3, \\ \mathrm{d}X_t^3 = X_t^3(\mu \mathrm{d}t + \sigma(\rho \mathrm{d}W_t^1 + \sqrt{1-\rho^2}\mathrm{d}W_t^2)) \end{cases},$$

where $g(t) = 0.4 + (0.1 + 0.2t)e^{-2t}$.

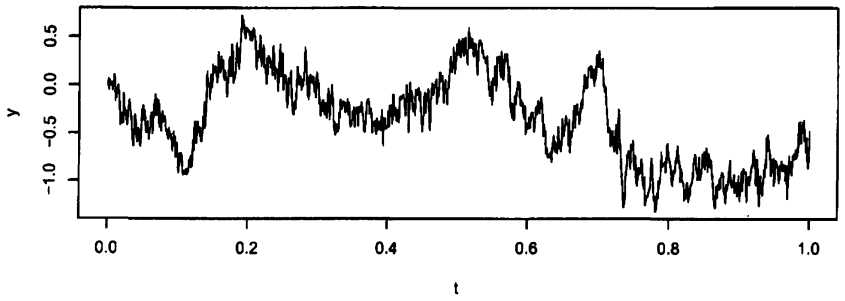
2.3.4 Fractional Gaussian noise

In order to specify a stochastic differential equation driven by fractional Gaussian noise it is necessary to specify the value of the Hurst parameter. For example, if we want to specify the following model

$$\mathrm{d}Y_t = 3Y_t\mathrm{d}t + \mathrm{d}W_t^H$$

we proceed as follows

```
R> mod4 <- setModel(drift = "3*y",
+     diffusion = 1, hurst = 0.3,
+     solve.var = "y")
R> set.seed(123)
R> X <- simulate(mod4, sampling = setSampling(n = 1000))
R> plot(X)
```



In this case, the appropriate slot is now filled

```
R> str(mod4)

Formal class 'yuima.model' [package "yuima"] with 16 slots
  ..@ drift          :  expression((3 * y))
  ..@ diffusion      :List of 1
  .. ..$ :  expression(1)
  ..@ hurst          : num 0.3
  ..@ jump.coeff     :  expression()
  ..@ measure        : list()
  ..@ measure.type   : chr(0)
  ..@ parameter      :Formal class 'model.parameter' [package "yuima"] with 6 slots
  .. .. ..@ all      : chr(0)
  .. .. ..@ common   : chr(0)
  .. .. ..@ diffusion: chr(0)
  .. .. ..@ drift    : chr(0)
  .. .. ..@ jump     : chr(0)
  .. .. ..@ measure  : chr(0)
  ..@ state.variable : chr "x"
  ..@ jump.variable  : chr(0)
  ..@ time.variable  : chr "t"
  ..@ noise.number   : num 1
  ..@ equation.number: int 1
  ..@ dimension      : int [1:6] 0 0 0 0 0 0
  ..@ solve.variable : chr "y"
  ..@ xinit          : num 0
  ..@ J.flag         : logi FALSE
```

2.3.5 Lévy processes

Jump processes can be specified in different ways in mathematics and hence in **yuima** package. Let Z_t be a Compound Poisson Process (i.e. jumps follow some distribution, like the Gaussian law). Then it is possible to consider the following SDE which involves jumps

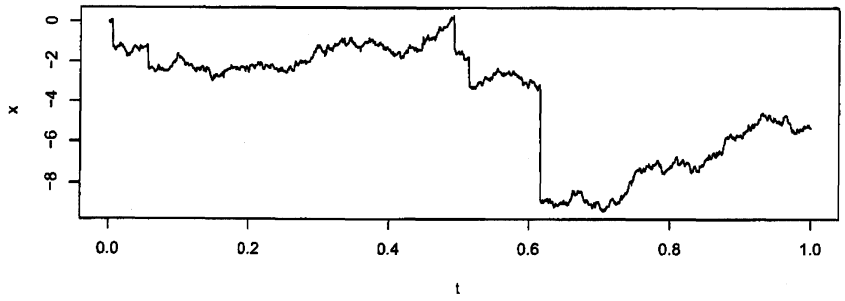
$$\mathrm{d}X_t = a(X_t)dt + b(X_t)\mathrm{d}W_t + \mathrm{d}Z_t$$

In the next example we consider a compound Poisson process with intensity $\lambda = 10$ with Gaussian jumps. This model can be specified in `setModel` using the argument `measure.type="CP"` A simple Ornstein-Uhlembeck process with Gaussian jumps

$$\mathrm{d}X_t = -\theta X_t\mathrm{d}t + \sigma\mathrm{d}W_t + Z_t$$

is specified as

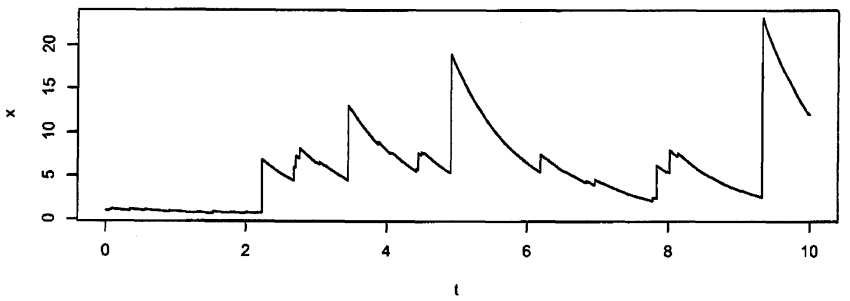
```
R> mod5 <- setModel(drift = c("-theta*x"),
+     diffusion = "sigma", jump.coeff = "1",
+     measure = list(intensity = "10",
+         df = list("dnorm(z, 0, 1)")),
+     measure.type = "CP", solve.variable = "x")
R> set.seed(123)
R> X <- simulate(mod5, true.p = list(theta = 1,
+     sigma = 3), sampling = setSampling(n = 1000))
R> plot(X)
```



Another possibility is to specify the Lévy measure. Without going into too much details, here is an example of specification of a simple Ornstein-Uhlembeck process with IG (Inverse Gaussian) Lévy measure

$$\mathrm{d}X_t = -x\mathrm{d}t + dZ_t$$

```
R> mod6 <- setModel(drift = "-x",
+     xinit = 1, jump.coeff = "1",
+     measure.type = "code", measure = list(df = "rIG(z, 1, 0.1)"))
R> set.seed(123)
R> X <- simulate(mod6, sampling = setSampling(Terminal = 10,
+     n = 10000))
R> plot(X)
```



2.3.6 Generic models

In general, the **yuima** package allows to specify a large family of models solutions to

$$\mathrm{d}X_t = a(X_t)\mathrm{d}t + b(X_t)\mathrm{d}W_t + c(X_t)\mathrm{d}Z_t$$

using the following interface

```
R> setModel(drift, diffusion, hurst = 0.5,
+     jump.coeff, measure, measure.type,
+     state.variable = "x", jump.variable = "z",
+     time.variable = "t", solve.variable,
+     xinit)
```

The **yuima** package implements many multivariate Random Numbers Generators (RNG) which are needed to simulate Lévy paths including `rIG` (Inverse Gaussian), `rNIG` (Normal Inverse Gaussian), `rbgamma` (Bilateral Gamma), `rngamma` (Gamma) and `rstable` (Stable Laws). Other user-defined RNG can be used freely.

3 Asymptotic expansion

The **yuima** package can handle asymptotic expansion of functionals of d-dimensional diffusion process

$$\mathrm{d}X_t^\varepsilon = a(X_t^\varepsilon, \varepsilon)\mathrm{d}t + b(X_t^\varepsilon, \varepsilon)\mathrm{d}W_t, \qquad \varepsilon \in (0, 1]$$

with W_t and r-dimensional Wiener process, i.e. $W_t = (W_t^1, \ldots, W_t^r)$. The functional is expressed in the following abstract form

$$F^\varepsilon(X_t^\varepsilon) = \sum_{\alpha=0}^{r} \int_0^T f_\alpha(X_t^\varepsilon, \mathrm{d})\mathrm{d}W_t^\alpha + F(X_t^\varepsilon, \varepsilon), \qquad W_t^0 = t$$

A typical example of application is the case of Asian option pricing. For example, in the Black & Scholes model

$$\mathrm{d}X_t^\varepsilon = \mu X_t^\varepsilon \mathrm{d}t + \varepsilon X_t^\varepsilon \mathrm{d}W_t$$

the price of the option is of the form

$$\mathbb{E}\left\{\max\left(\frac{1}{T}\int_0^T X_t^\varepsilon \mathrm{d}t - K, 0\right)\right\}.$$

Thus the functional of interest is

$$F^\varepsilon(X_t^\varepsilon) = \frac{1}{T}\int_0^T X_t^\varepsilon \mathrm{d}t, \qquad r = 1$$

with

$$f_0(x,\varepsilon) = \frac{x}{T}, \quad f_1(x,\varepsilon) = 0, \quad F(x,\varepsilon) = 0$$

in

$$F^\varepsilon(X_t^\varepsilon) = \sum_{\alpha=0}^{r} \int_0^T f_\alpha(X_t^\varepsilon, \mathrm{d})\mathrm{d}W_t^\alpha + F(X_t^\varepsilon, \varepsilon)$$

So, the call option price requires the composition of a smooth functional

$$F^\varepsilon(X_t^\varepsilon) = \frac{1}{T}\int_0^T X_t^\varepsilon \mathrm{d}t, \qquad r = 1$$

with the irregular function

$$\max(x - K, 0)$$

Monte Carlo methods require a huge number of simulations to get the desired accuracy of the calculation of the price, while asymptotic expansion of F^ε provides very accurate approximations. The **yuima** package provides functions to construct the functional F^ε, and automatic asymptotic expansion based on Malliavin calculus starting from a **yuima** object. Next is an example

```
R> diff.matrix <- matrix(c("x*e"),
+     1, 1)
R> model <- setModel(drift = c("x"),
+     diffusion = diff.matrix)
R> T <- 1
R> xinit <- 1
R> K <- 1
R> f <- list(expression(x/T), expression(0))
R> F <- 0
R> e <- 0.3
R> yuima <- setYuima(model = model,
+     sampling = setSampling(Terminal = T,
+         n = 1000))
R> yuima <- setFunctional(yuima, f = f,
+     F = F, xinit = xinit, e = e)
```

this time the `setFunctional` command fills the appropriate slots

```
R> str(yuima@functional)
```

```
Formal class 'yuima.functional' [package "yuima"] with 4 slots
  ..@ F    : num 0
  ..@ f    :List of 2
  .. ..$ : expression(x/T)
  .. ..$ : expression(0)
  ..@ xinit: num 1
  ..@ e    : num 0.3
```

Then, it is as easy as

```
R> F0 <- F0(yuima)
R> F0

[1] 1.717423

R> max(F0 - K, 0)

[1] 0.7174228
```

to obtain the zero order approximation of the value of the functional. We can go up to the first order approximation adding one term to the expansion

```
R> rho <- expression(0)
R> get_ge <- function(x, epsilon,
+     K, F0) {
+     tmp <- (F0 - K) + (epsilon *
+         x)
+     tmp[(epsilon * x) < (K - F0)] <- 0
+     return(tmp)
+ }
R> epsilon <- e
R> g <- function(x) {
+     tmp <- (F0 - K) + (epsilon *
+         x)
+     tmp[(epsilon * x) < (K - F0)] <- 0
+     tmp
+ }
R> asymp <- asymptotic_term(yuima,
+     block = 10, rho, g)
```

and the final value is

```
R> asymp$d0 + e * asymp$d1

[1] 0.7158789
```

4 Quasi Maximum Likelihood estimation

Consider the multidimensional diffusion process

$$\mathrm{d}X_t = b(\theta_2, X_t)\mathrm{d}t + \sigma(\theta_1, X_t)\mathrm{d}W_t$$

where W_t is an r-dimensional standard Wiener process independent of the initial value $X_0 = x_0$. Quasi-MLE assumes the following approximation of the true log-likelihood for multidimensional diffusions

$$\ell_n(\mathbf{X}_n, \theta) = -\frac{1}{2}\sum_{i=1}^{n}\left\{\log\det(\Sigma_{i-1}(\theta_1)) + \frac{1}{\Delta_n}\Sigma_{i-1}^{-1}(\theta_1)[\Delta X_i - \Delta_n b_{i-1}(\theta_2)]^{\otimes 2}\right\} \quad (4.1)$$

where $\theta = (\theta_1, \theta_2)$, $\Delta X_i = X_{t_i} - X_{t_{i-1}}$, $\Sigma_i(\theta_1) = \Sigma(\theta_1, X_{t_i})$, $b_i(\theta_2) = b(\theta_2, X_{t_i})$, $\Sigma = \sigma^{\otimes 2}$, $A^{\otimes 2} = A^T A$ and A^{-1} the inverse of A, $A[B]^{\otimes 2} = B^T AB$. Then, Yoshida (1992), the QML estimator of θ is

$$\tilde{\theta}_n = \arg\min_{\theta} \ell_n(\mathbf{X}_n, \theta)$$

As an example, we consider the simple model

$$\mathrm{d}X_t = -\theta_2 X_t \mathrm{d}t + \theta_1 \mathrm{d}W_t \quad (4.2)$$

with $\theta_1 = 0.3$ and $\theta_2 = 0.1$

```
R> ymodel <- setModel(drift = "-x*theta2",
+     diffusion = "theta1", time.variable = "t",
+     state.variable = "x", solve.variable = "x")
R> n <- 1000
R> ysamp <- setSampling(Terminal = (n)^(1/3),
+     n = n)
R> yuima <- setYuima(model = ymodel,
+     sampling = ysamp)
R> set.seed(123)
R> yuima <- simulate(yuima, xinit = 1,
+     true.parameter = list(theta1 = 0.3,
+         theta2 = 0.1))
```

With the simulated path we can use the function `qmle` to estimate the parameters as follows

```
R> mle1 <- qmle(yuima, start = list(theta1 = 0.8,
+     theta2 = 0.7), lower = list(theta1 = 0.05,
+     theta2 = 0.05), upper = list(theta1 = 0.5,
+     theta2 = 0.5), method = "L-BFGS-B")
```

and the estimated coefficients are as follows

```
R> coef(mle1)

   theta1    theta2
0.3015202 0.1029822
```

```
R> summary(mle1)

Maximum likelihood estimation

Call:
qmle(yuima = yuima, start = list(theta1 = 0.8, theta2 = 0.7),
    method = "L-BFGS-B", lower = list(theta1 = 0.05, theta2 = 0.05),
    upper = list(theta1 = 0.5, theta2 = 0.5))

Coefficients:
        Estimate  Std. Error
theta1 0.3015202 0.006879348
theta2 0.1029822 0.114539931

-2 log L: -4192.279
```

5 Adaptive Bayes estimation

Consider again the diffusion process solution to

$$\mathrm{d}X_t = b(X_t, \theta_2)\mathrm{d}t + \sigma(X_t, \theta_1)\mathrm{d}W_t, \tag{5.1}$$

and the quasi likelihood defined in (4.1).

The adaptive Bayes type estimator is defined as follows. First we choose an initial arbitrary value $\theta_2^\star \in \Theta_2$ and pretend θ_1 is the unknown parameter to make the Bayesian type estimator $\tilde{\theta}_1$ as

$$\tilde{\theta}_1 = \Big[\int_{\Theta_1} \ell_n(\mathbf{x}_n, (\theta_1, \theta_2^\star))\pi_1(\theta_1)d\theta_1\Big]^{-1} \int_{\Theta_1} \theta_1 \ell_n(\mathbf{x}_n, (\theta_1, \theta_2^\star))\pi_1(\theta_1)d\theta_1 \tag{5.2}$$

where π_1 is a prior density on Θ_1. According to the asymptotic theory, if π_1 is positive on Θ_1, any function can be used. For estimation of θ_2, we use $\tilde{\theta}_1$ to reform the quasi-likelihood function. That is, the Bayes type estimator for θ_2 is defined by

$$\tilde{\theta}_2 = \Big[\int_{\Theta_2} \ell_n(\mathbf{x}_n, (\tilde{\theta}_1, \theta_2))\pi_2(\theta_2)d\theta_2\Big]^{-1} \int_{\Theta_2} \theta_2 \ell_n(\mathbf{x}_n, (\tilde{\theta}_1, \theta_2))\pi_2(\theta_2)d\theta_2 \tag{5.3}$$

where π_2 is a prior density on Θ_2. In this way, we obtain the adaptive Bayes type estimator $\tilde{\theta} = (\tilde{\theta}_1, \tilde{\theta}_2)$ for $\theta = (\theta_1, \theta_2)$.

Adaptive Bayes estimation is developed in **yuima** via the method `adaBayes`. Consider again the model (4.2) with the same values for the parameters, i.e. $\theta_1 = 0.3$ and $\theta_2 = 0.1$ In order to perform Bayesian estimation, we need to prepare the prior densities for the parameters. For simplicity we use uniform distributions in $[0, 1]$

```
R> prior <- list(theta2 = list(measure.type = "code",
+     df = "dunif(z,0,1)"), theta1 = list(measure.type = "code",
+     df = "dunif(z,0,1)"))
```

Then we call adaBayes as follows

```
R> param.init <- list(theta2 = 0.5,
+     theta1 = 0.5)
R> bayes1 <- adaBayes(yuima, start = param.init,
+     prior = prior, method = "nomcmc")
```

and we can compare the adaptive Bayes estimates with the QMLE estimates

```
R> bayes1@coef
```

```
   theta1    theta2
0.2996045 0.1629653
```

```
R> coef(mle1)
```

```
   theta1    theta2
0.3015202 0.1029822
```

The argument method="nomcmc" in adaBayes performs numerical integration, otherwise MCMC method is used.

6 Asynchronous covariance estimation

Suppose that two Itô processes are observed only at discrete times in a non-synchronous manner. We are interested in estimating the covariance of the two processes accurately in such a situation. This type of problem arises typically in high-frequency financial time series.

Let $T \in (0, \infty)$ be a terminal time for possible observations. We consider a two dimensional Itô process (X^1, X^2) satisfying the stochastic differential equations

$$\begin{aligned} \mathrm{d}X_t^l &= \mu_t^l \mathrm{d}t + \sigma_t^l \mathrm{d}W_t^l, \quad t \in [0, T] \\ X_0^l &= x_0^l \end{aligned}$$

for $l = 1, 2$. Here W^l denote standard Wiener processes with a progressively measurable correlation process $\mathrm{d}\langle W_1, W_2 \rangle_t = \rho_t \mathrm{d}t$, μ_t^l and σ_t^l are progressively measurable processes, and x_0^l are initial random variables independent of (W^1, W^2). Diffusion type processes are in the scope but this model can express more sophisticated stochastic structures.

The process X^l is supposed to be observed at over the increasing sequence of times $T^{l,i}$ ($i \in \mathbb{Z}_{\geq 0}$) starting at 0, up to time T. Thus, the observables are $(T^{l,i}, X^{l,i})$ with $T^{l,i} \leq T$. Each $T^{l,j}$ may be a stopping time, so possibly depends on the history of (X^1, X^2) as well as the precedent stopping times. Two sequences of stopping times $T^{1,i}$ and $T^{2,j}$ are *nonsynchronous*, and irregularly spaced, in general. In particular, cce can apply to estimation of the quadratic variation of a single stochastic process sampled regularly/irregularly.

The parameter of interest is the quadratic covariation between X^1 and X^2:

$$\theta = \langle X^1, X^2 \rangle_T = \int_0^T \sigma_t^1 \sigma_t^2 \rho_t \mathrm{d}t. \tag{6.1}$$

The target variable θ is random in general.

It can be estimated with the nonsynchronous covariance estimator (Hayashi-Yoshida estimator)

$$U_n = \sum_{i,j:T^{1,i} \leq T, T^{2,j} \leq T} (X^1_{T^{1,i}} - X^1_{T^{1,i-1}})(X^2_{T^{2,j}} - X^2_{T^{2,j-1}}) 1_{\{(T^{1,i-1}, T^{1,i}] \bigcap (T^{2,j-1}, T^{2,j}] \neq \emptyset\}}. \tag{6.2}$$

That is, the product of any pair of increments $(X^1_{T^{1,i}} - X^1_{T^{1,i-1}})$ and $(X^2_{T^{2,j}} - X^2_{T^{2,j-1}})$ will make a contribution to the sum only when the respective observation intervals $(T^{1,i-1}, T^{1,i}]$ and $(T^{2,j-1}, T^{2,j}]$ are overlapping with each other. It is known that U_n is consist and has asymptotically mixed normal distribution as $n \to \infty$ if the maximum length between two consecutive observing times tends to 0. See Hayashi and Yoshida (2005, 2008a, 2006, 2008b) for details.

6.1 Example: data generation and estimation by yuima package

We will demonstrate how to apply cce function to nonsynchronous high-frequency data by simulation. As an example, consider a two dimensional stochastic process (X_t^1, X_t^2) satisfying the stochastic differential equation

$$\begin{aligned} \mathrm{d}X_t^1 &= \sigma_{1,t} \mathrm{d}B_t^1, \\ \mathrm{d}X_t^2 &= \sigma_{2,t} \mathrm{d}B_t^2. \end{aligned} \tag{6.3}$$

Here B_t^1 and B_t^2 denote two standard Wiener processes, however they are correlated as

$$B_t^1 = W_t^1, \tag{6.4}$$

$$B_t^2 = \int_0^t \rho_s \mathrm{d}W_s^1 + \int_0^t \sqrt{1 - \rho_s^2} \mathrm{d}W_s^2, \tag{6.5}$$

where W_t^1 and W_t^2 are independent Wiener processes, and ρ_t is the correlation function between B_t^1 and B_t^2. We consider $\sigma_{l,t}, l = 1, 2$ and ρ_t of the following form in this example:

$$\begin{aligned} \sigma_{1,t} &= \sqrt{1+t}, \\ \sigma_{2,t} &= \sqrt{1+t^2}, \\ \rho_t &= \frac{1}{\sqrt{2}}. \end{aligned}$$

To simulate the stochastic process (X_t^1, X_t^2), we first build the model by `setModel` as before. It should be noted that the method of generating non-synchronous data can be replaced by a simpler one but we will take a general approach here to demonstrate a usage of the **yuima** comprehensive package for simulation and estimation of stochastic processes.

```
R> diff.coef.1 <- function(t, x1 = 0,
+     x2 = 0) sqrt(1 + t)
R> diff.coef.2 <- function(t, x1 = 0,
+     x2 = 0) sqrt(1 + t^2)
R> cor.rho <- function(t, x1 = 0,
+     x2 = 0) sqrt(1/2)
R> diff.coef.matrix <- matrix(c("diff.coef.1(t,x1,x2)",
+     "diff.coef.2(t,x1,x2) * cor.rho(t,x1,x2)",
+     "", "diff.coef.2(t,x1,x2) * sqrt(1-cor.rho(t,x1,x2)^2)"),
+     2, 2)
R> cor.mod <- setModel(drift = c("",
+     ""), diffusion = diff.coef.matrix,
+     solve.variable = c("x1", "x2"))
```

The parameter we want to estimate is the quadratic covariation between X_1 and X_2:

$$\theta = \langle X_1, X_2 \rangle_T = \int_0^T \sigma_{1,t}\sigma_{2,t}\rho_t \mathrm{d}t. \tag{6.6}$$

Later, we will compare estimated values with the true value of θ given by

```
R> CC.theta <- function(T, sigma1,
+     sigma2, rho) {
+     tmp <- function(t) return(sigma1(t) *
+         sigma2(t) * rho(t))
+     integrate(tmp, 0, T)
+ }
```

For the sampling scheme, we will consider the independent Poisson sampling. That is, each configuration of the sampling times $T^{l,i}$ is realized as the Poisson random measure with intensity np_l, and the two random measures are

independent each other as well as the stochastic processes. Then it is known from today's lecture that

$$n^{1/2}(U_n - \theta) \to N(0, c), \tag{6.7}$$

as $n \to \infty$, where

$$c = \left(\frac{2}{p_1} + \frac{2}{p_2}\right) \int_0^T (\sigma_{1,t}\sigma_{2,t})^2 \, \mathrm{d}t + \left(\frac{2}{p_1} + \frac{2}{p_2} - \frac{2}{p_1 + p_2}\right) \int_0^T (\sigma_{1,t}\sigma_{2,t}\rho_t)^2 \, \mathrm{d}t. \tag{6.8}$$

```
R> set.seed(123)
R> Terminal <- 1
R> n <- 1000
R> theta <- CC.theta(T = Terminal,
+     sigma1 = diff.coef.1, sigma2 = diff.coef.2,
+     rho = cor.rho)$value
R> cat(sprintf("theta=%5.3f\n", theta))

theta=1.000
```

so in our case $\theta = 1$.

```
R> yuima.samp <- setSampling(Terminal = Terminal,
+     n = n)
R> yuima <- setYuima(model = cor.mod,
+     sampling = yuima.samp)
R> X <- simulate(yuima)
```

cce takes the sample and returns an estimate of the quadratic covariation. For example, for the complete data

```
R> cce(X)

$covmat
         [,1]     [,2]
[1,] 1.491938 1.086078
[2,] 1.086078 1.474730

$cormat
          [,1]      [,2]
[1,] 1.0000000 0.7321992
[2,] 0.7321992 1.0000000
```

and we now apply random sampling

```
R> p1 <- 0.2
R> p2 <- 0.3
R> newsamp <- setSampling(random = list(rdist = c(function(x) rexp(x,
+     rate = p1 * n/Terminal), function(x) rexp(x,
+     rate = p1 * n/Terminal))))
R> Y <- subsampling(X, sampling = newsamp)
R> cce(Y)

$covmat
         [,1]     [,2]
[1,] 1.397269 1.070313
[2,] 1.070313 1.338464

$cormat
          [,1]      [,2]
[1,] 1.0000000 0.7826494
[2,] 0.7826494 1.0000000
```

Now we calculate the asymptotic variance of the estimator using (6.8)

```
R> var.c <- function(T, p1, p2, sigma1,
+     sigma2, rho) {
+     tmp_integrand1 <- function(t) (sigma1(t) *
+         sigma2(t))^2
+     i1 <- integrate(tmp_integrand1,
+         0, T)
+     tmp_integrand2 <- function(t) (sigma1(t) *
+         sigma2(t) * rho(t))^2
+     i2 <- integrate(tmp_integrand2,
+         0, T)
+     2 * (1/p1 + 1/p2) * i1$value +
+         2 * (1/p1 + 1/p2 - 1/(p1 +
+             p2)) * i2$value
+ }
R> vc <- var.c(T = Terminal, p1, p2,
+     diff.coef.1, diff.coef.2, cor.rho)
R> sqrt(vc/n)

[1] 0.2188988
```

7 Change point analysis

Consider a multidimensional stochastic differential equation of the form

$$\mathrm{d}Y_t = b_t \mathrm{d}t + \sigma(X_t, \theta)\mathrm{d}W_t, \quad t \in [0, T],$$

where W_t a r-dimensional Wiener process and b_t and X_t are multidimensional processes and σ is the diffusion coefficient (volatility) matrix. When $Y = X$ the problem is a diffusion model. The process b_t may have jumps but should not explode and it is treated as a nuisance in this model. The change-point problem for the volatility is formalized as follows

$$Y_t = \begin{cases} Y_0 + \int_0^t b_s ds + \int_0^t \sigma(X_s, \theta_1^*) dW_s & \text{for } t \in [0, \tau^*) \\ Y_{\tau^*} + \int_{\tau^*}^t b_s ds + \int_{\tau^*}^t \sigma(X_s, \theta_2^*) dW_s & \text{for } t \in [\tau^*, T]. \end{cases}$$

The change point τ^* instant is unknown and is to be estimated, along with θ_1^* and θ_2^*, from the observations sampled from the path of (X, Y). The **yuima** implements the quasi-maximum likelihood approach as in Iacus and Yoshida (2009) described in the following. Let $\Delta_i Y = Y_{t_i} - Y_{t_{i-1}}$ and define

$$\Phi_n(t; \theta_1, \theta_2) = \sum_{i=1}^{[nt/T]} G_i(\theta_1) + \sum_{i=[nt/T]+1}^{n} G_i(\theta_2), \tag{7.1}$$

with

$$G_i(\theta) = \log \det S(X_{t_{i-1}}, \theta) + \Delta_n^{-1} (\Delta_i Y)' S(X_{t_{i-1}}, \theta)^{-1} (\Delta_i Y). \tag{7.2}$$

Suppose that there exists an estimator $\hat{\theta}_k$ for each θ_k, $k = 1, 2$. In case θ_k^* are known, we define $\hat{\theta}_k$ just as $\hat{\theta}_k = \theta_k^*$. The change point estimator of τ^* is

$$\hat{\tau}_n = \underset{t \in [0,T]}{\arg\min}\, \Phi_n(t; \hat{\theta}_1, \hat{\theta}_2).$$

7.1 Example of Volatility Change-Point Estimation

Consider the 2-dimensional stochastic differential equation

$$\begin{pmatrix} \mathrm{d}X_t^1 \\ \mathrm{d}X_t^2 \end{pmatrix} = \begin{pmatrix} 1 - X_t^1 \\ 3 - X_t^2 \end{pmatrix} \mathrm{d}t + \begin{bmatrix} \theta_{1.1} \cdot X_t^1 & 0 \cdot X_t^1 \\ 0 \cdot X_t^2 & \theta_{1.2} \cdot X_t^2 \end{bmatrix}' \begin{pmatrix} \mathrm{d}W_t^1 \\ \mathrm{d}W_t^2 \end{pmatrix}$$

$$X_0^1 = 1.0, \quad X_0^2 = 1.0,$$

with change point instant at time $\tau = 0.4$. Some code is needed to simulate such a process. First we define the model

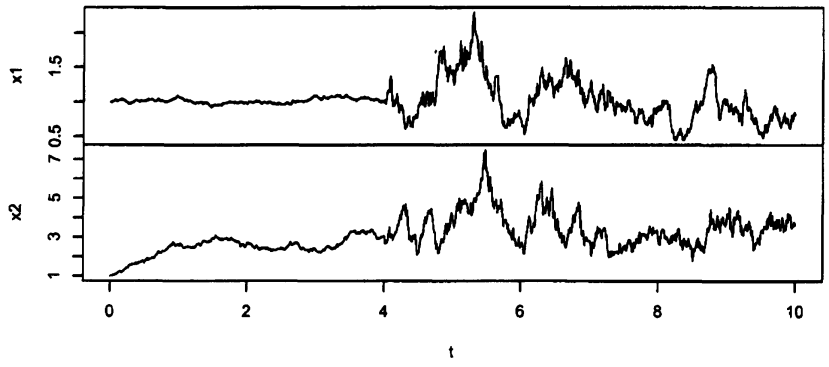
```
R> diff.matrix <- matrix(c("theta1.1*x1",
+     "0*x2", "0*x1", "theta1.2*x2"),
+     2, 2)
R> drift.c <- c("1-x1", "3-x2")
R> drift.matrix <- matrix(drift.c,
+     2, 1)
R> ymodel <- setModel(drift = drift.matrix,
+     diffusion = diff.matrix, time.variable = "t",
+     state.variable = c("x1", "x2"),
+     solve.variable = c("x1", "x2"))
```

and then simulate two trajectories. One up to the change point $\tau = 4$ with parameters $\theta_{1.1} = 0.1$ and $\theta_{1.1} = 0.2$, and a second trajectory with parameters $\theta_{1.1} = 0.6$ and $\theta_{1.2} = 0.6$. For the second trajectory, the initial value is set to the last value of the first trajectory.

```
R> n <- 1000
R> set.seed(123)
R> t1 <- list(theta1.1 = 0.1, theta1.2 = 0.2)
R> t2 <- list(theta1.1 = 0.6, theta1.2 = 0.6)
R> tau <- 0.4
R> ysamp1 <- setSampling(n = tau *
+     n, Initial = 0, delta = 0.01)
R> yuima1 <- setYuima(model = ymodel,
+     sampling = ysamp1)
R> yuima1 <- simulate(yuima1, xinit = c(1,
+     1), true.parameter = t1)
R> x1 <- yuima1@data@zoo.data[[1]]
R> x1 <- as.numeric(x1[length(x1)])
R> x2 <- yuima1@data@zoo.data[[2]]
R> x2 <- as.numeric(x2[length(x2)])
R> ysamp2 <- setSampling(Initial = n *
+     tau * 0.01, n = n * (1 - tau),
+     delta = 0.01)
R> yuima2 <- setYuima(model = ymodel,
+     sampling = ysamp2)
R> yuima2 <- simulate(yuima2, xinit = c(x1,
+     x2), true.parameter = t2)
R> yuima <- yuima1
R> yuima@data@zoo.data[[1]] <- c(yuima1@data@zoo.data[[1]],
+     yuima2@data@zoo.data[[1]][-1])
R> yuima@data@zoo.data[[2]] <- c(yuima1@data@zoo.data[[2]],
+     yuima2@data@zoo.data[[2]][-1])
```

The composed trajectory appears as follows

```
R> plot(yuima)
```



Just as an example, we test the ability of the change point estimator to identify τ when for given true values of the parameters $\theta_{1.1}$ and $\theta_{1.2}$

```
R> t.est <- CPoint(yuima, param1 = t1,
+     param2 = t2, plot = TRUE)
R> t.est$tau
```

```
[1] 3.99
```

A two stage change point estimation approach is available as explained in Iacus and Yoshida (2009).

8 LASSO model selection

Let X_t be a diffusion process solution to

$$\mathrm{d}X_t = b(\alpha, X_t)\mathrm{d}t + \sigma(\beta, X_t)\mathrm{d}W_t$$

$$\alpha = (\alpha_1, ..., \alpha_p)' \in \Theta_p \subset \mathbb{R}^p, \quad p \geq 1$$

$$\beta = (\beta_1, ..., \beta_q)' \in \Theta_q \subset \mathbb{R}^q, \quad q \geq 1$$

with $b : \Theta_p \times \mathbb{R}^d \to \mathbb{R}^d$, $\sigma : \Theta_q \times \mathbb{R}^d \to \mathbb{R}^d \times \mathbb{R}^m$ and $W_t, t \in [0, T]$, is a standard Brownian motion in $\mathbb{R}^m$. We assume that the functions b and σ are known up to α and β. We denote by $\theta = (\alpha, \beta) \in \Theta_p \times \Theta_q = \Theta$ the parametric vector and with $\theta_0 = (\alpha_0, \beta_0)$ its unknown true value. Let $\mathbb{H}_n(\mathbf{X}_n, \theta) = \ell_n(\mathbf{X}_n, \theta)$ from equation (4.1). The quasi-MLE $\tilde{\theta}_n$ for this model is the solution of the following problem

$$\tilde{\theta}_n = (\tilde{\alpha}_n, \tilde{\beta}_n)' = \arg\min_{\theta} \mathbb{H}_n(\mathbf{X}_n, \theta)$$

The adaptive LASSO estimator is defined as the solution to the quadratic problem under L_1 constraints

$$\hat{\theta}_n = (\hat{\alpha}_n, \hat{\beta}_n) = \arg\min_{\theta} \mathcal{F}(\theta).$$

with

$$\mathcal{F}(\theta) = (\theta - \tilde{\theta}_n)\ddot{\mathbb{H}}_n(\mathbf{X}_n, \tilde{\theta}_n)(\theta - \tilde{\theta}_n)' + \sum_{j=1}^{p} \lambda_{n,j}|\alpha_j| + \sum_{k=1}^{q} \gamma_{n,k}|\beta_k|$$

For more details see De Gregorio and Iacus (2010). The tuning parameters should be chosen as in Zou (2006) in the following way

$$\lambda_{n,j} = \lambda_0 |\tilde{\alpha}_{n,j}|^{-\delta_1}, \qquad \gamma_{n,k} = \gamma_0 |\tilde{\beta}_{n,j}|^{-\delta_2} \tag{8.1}$$

where $\tilde{\alpha}_{n,j}$ and $\tilde{\beta}_{n,k}$ are the unpenalized QML estimator of α_j and β_k respectively, $\delta_1, \delta_2 > 0$ and usually taken unitary.

8.1 An example of use

The `lasso` method is implemented in the **yuima** package. Let us consider the full CKLS model

$$\mathrm{d}X_t = (\alpha + \beta X_t)\mathrm{d}t + \sigma X_t^{\gamma}\mathrm{d}W_t$$

and let us try to estimate the parameter on the U.S. Interest Rates monthly data from 06/1964 to 12/1989. We prepare the data, the model and the constraints for optimization

```
R> library(Ecdat)
R> data(Irates)
R> rates <- Irates[, "r1"]
R> plot(rates)
R> X <- window(rates, start = 1964.471,
+     end = 1989.333)
R> mod <- setModel(drift = "alpha+beta*x",
+     diffusion = matrix("sigma*x^gamma",
+         1, 1))
R> yuima <- setYuima(data = setData(X),
+     model = mod)
R> lambda10 <- list(alpha = 10, beta = 10,
+     sigma = 10, gamma = 10)
R> start <- list(alpha = 1, beta = -0.1,
+     sigma = 0.1, gamma = 1)
R> low <- list(alpha = -5, beta = -5,
+     sigma = -5, gamma = -5)
R> upp <- list(alpha = 8, beta = 8,
+     sigma = 8, gamma = 8)
```

and now we apply the `lasso` function

```
R> lasso10 <- lasso(yuima, lambda10,
+     start = start, lower = low,
+     upper = upp, method = "L-BFGS-B")
```

From which we see that, instead of the general model

$$\mathrm{d}X_t = (\alpha + \beta X_t)\mathrm{d}t + \sigma X_t^{\gamma}\mathrm{d}W_t$$

```
R> round(lasso10$mle, 2)

sigma gamma alpha  beta
 0.13  1.44  2.08 -0.26

R> round(lasso10$lasso, 2)

sigma gamma alpha  beta
 0.12  1.50  0.59  0.00
```

the LASSO method selects the reduced model

$$\mathrm{d}X_t = 0.6\mathrm{d}t + 0.12X_t^{\frac{3}{2}}\mathrm{d}W_t$$

Acknowledgements

The author thanks all the members of the Yuima Project Team. All the errors in this paper are solely of the present author.

References

Chambers, J. M. (1998). *Programming with Data: A Guide to the S Language.* Springer-Verlag, New York.

De Gregorio, A. and Iacus, S. M. (2010). Adaptive lasso-type estimation for ergodic diffusion processes. *http://services.bepress.com/unimi/statistics/art50/* .

Hayashi, T. and Yoshida, N. (2005). On covariance estimation of non-synchronously observed diffusion processes. *Bernoulli* **11**, 359–379.

Hayashi, T. and Yoshida, N. (2006). Nonsynchronous covariance estimator and limit theorem. *Institute of Statistical Mathematics* **Research Memorandum No.1020**, 1–40.

Hayashi, T. and Yoshida, N. (2008a). Asymptotic normality of a covariance estimator for nonsynchronously observed diffusion processes. *Annals of the Institute of Statistical Mathematics* **60**, 367–406.

Hayashi, T. and Yoshida, N. (2008b). Nonsynchronous covariance estimator and limit theorem ii. *Institute of Statistical Mathematics* **Research Memorandum No.1067**, 1–40.

Iacus, S. and Yoshida, N. (2009). Estimation for the change point of the volatility in a stochastic differential equation. *http://arxiv.org/abs/0906.3108* .

Yoshida, N. (1992). Estimation for diffusion processes from discrete observation. *J. Multivar. Anal.* **41**, 2, 220–242.

Zou, H. (2006). The adaptive lasso and its oracle properties. *J. Amer. Stat. Assoc.* **101**, 476, 1418–1429.

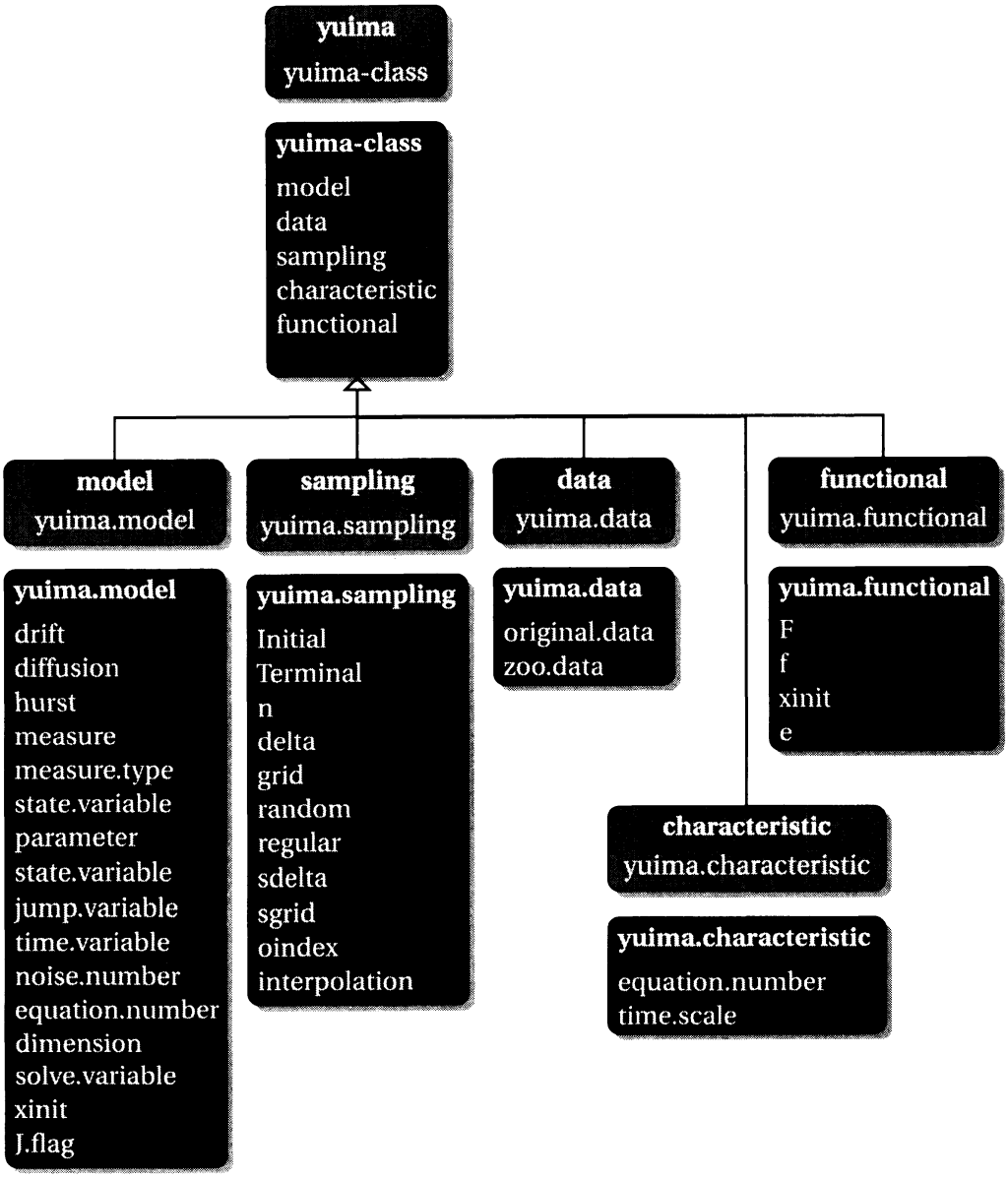


Figure 1: The main classes in the **yuima** package.